

⑨ 日本国特許庁(JP)

⑩ 特許出願公開

⑫ 公開特許公報(A)

昭62-264333

⑮ Int.Cl.⁴

G 06 F 9/44

識別記号

3 2 2

庁内整理番号

A-8120-5B

⑬ 公開 昭和62年(1987)11月17日

審査請求 未請求 発明の数 1 (全5頁)

⑭ 発明の名称 プログラム最適化処理方法

⑯ 特 願 昭61-109107

⑰ 出 願 昭61(1986)5月13日

⑱ 発 明 者 木 村 浩 三 門真市大字門真1006番地 松下電器産業株式会社内

⑲ 出 願 人 松下電器産業株式会社 門真市大字門真1006番地

⑳ 代 理 人 弁理士 中尾 敏男 外1名

明 細 書

1、発明の名称

プログラム最適化処理方法

2、特許請求の範囲

ソース・プログラム中のプリプロセッサ・コントロール行を処理するマクロ処理部と、前記ソース・プログラム中でインラインコード展開させる関数を指定する関数指定部と、前記関数指定部で指定された関数を前記マクロ処理部の出力プログラム中でインラインコードに展開する関数処理部と、前記関数処理部の出力プログラムをコンパイル処理してオブジェクトプログラムを生成するコンパイル処理部とを備えていることを特徴とするプログラム最適化処理方法。

3、発明の詳細な説明

産業上の利用分野

本発明は、データ処理システムにおけるプログラムの最適化処理方法に関し、特にソース・プログラム中の指定する関数について関数呼び出し部分に関数定義部分(関数の宣言、手続きを記述し

た本体部分)をインラインコード展開機能を持つことより、オブジェクト・プログラムの実行を高速化できるプログラム最適化処理方法に関する。

従来の技術

従来のプログラム処理方法としては、例えば産業図書 中田育男著 コンパイラ 2章コンパイラの構造に記載されている。

第5図は入力をC言語(C言語の文法は共立出版 B.W.カーニハン D.M.リッチー著 プログラミング言語Cに準拠)ソース・プログラムとして従来のプログラム処理方法の構成図である。第5図において、9はC言語でかかれたソース・プログラム、10はC言語ソース・プログラム9中のプリプロセッサ・コントロール行を処理するマクロ処理部、11はマクロ処理部10でC言語ソース・プログラム9をマクロ処理された出力であるC言語中間プログラム、12はC言語中間プログラム11をコンパイルレオブジェクト・プログラムを生成するコンパイル処理部、13はコンパイル処理部12で生成されたオブジェクト・プ

プログラムである。

第6図は第5図の動作説明を行うために挙げた第5図のC言語ソース・プログラム9の一例である。

第7図は第5図のC言語中間プログラム11の一例である。

第8図は第6図同様、第5図のC言語ソース・プログラム9の一例である。

以上のように構成された従来のプログラム処理方法において、第6図を第5図のC言語ソース・プログラム9の例とした時の動作を第1の従来例として説明する。

第6図をC言語ソース・プログラム9としてマクロ処理部10に入力する。マクロ処理部10は第7図をC言語中間プログラム11として出力する。つまりマクロ処理部はメインルーチン中のSUBR(X)を(X&3)に置換する。コンパイル処理部12はC言語中間プログラム11を入力し、オブジェクトプログラム13を出力する。

次に第8図を第5図のC言語ソース・プログラ

ムが遅いという問題点も有していた。

本発明はかかる点に鑑み、指定する関数の呼び出し部分のみ関数定義部分をインラインコード展開可能なプログラム最適化処理方法を提供することを目的とする。

問題点を解決するための手段

本発明は、インラインコード展開させる関数呼び出し部分を指定する関数指定部と指定された関数呼び出し部分に関数定義部分をインラインコード展開する関数処理部を備えたプログラム最適化処理方法である。

作 用

本発明は前記した構成により、関数指定部が指定した関数呼び出し部分について関数処理部が関数定義部分をインラインコード展開すると、小さな記憶容量の増加でオブジェクト・プログラムを高速に実行できる。

実 施 例

第1図は本発明の実施例におけるプログラム最適化処理方法の構成図を示すものである。第1図

ム9の例とした時の動作を第2の従来例として説明する。

第8図をC言語ソース・プログラム9としてマクロ処理部10に入力する。マクロ処理部10は入力と同じ第8図をC言語中間プログラム11として出力する。コンパイル処理部12はC言語中間プログラム11を入力し、オブジェクトプログラム13を出力する。

発明が解決しようとする問題点

しかしながら上記のような構成では、ある一つの関数呼び出し部分については第6図、第7図のようにマクロ処理されてしまう、もしくは第8図のようにサブルーチンとして呼ばれる。以上の一方の形しか選択できなかった。このためすべての関数呼び出し部分をマクロ処理すると大きな記憶容量を必要とし、逆にマクロ処理させる関数の大きさにも制限が発生するという問題点を有していた。同様に関数がサブルーチンとして頻繁に呼ばれた場合には、オブジェクト・プログラム13の実行は関数の呼び出しによる不連続となり実行速

において、1はC言語ソース・プログラム、2はC言語ソース・プログラム1中のプリプロセッサ・コントロール行を処理するマクロ処理部、3はインラインコード展開を行う関数を指定する関数指定部、4は関数指定部3で指定された関数について関数呼び出し部分に関数定義部分をインラインコード展開を行う関数処理部、5は関数処理部4でインラインコード展開されたC言語中間プログラム、6はC言語中間プログラム5をコンパイルレオブジェクト・プログラムを生成するコンパイル処理部、7はコンパイル処理部6で生成されたオブジェクト・プログラムである。

第2図は、第1図の関数指定部3について詳しく説明するための図であり、8はC言語ソース・プログラム1中で指定したい関数のラインナンバーと関数名を書き込む関数指定テーブルである。

第3図は、第1図の動作説明を行なうために挙げた第1図のC言語ソース・プログラム1の例である。

第4図は、第1図のC言語中間プログラム5の

例である。

まず、以上のように構成された本実施例のプログラム最適化処理方法について第3図、第4図を第1図のそれぞれC言語ソース・プログラム1、C言語中間プログラム6の例とした時の動作を説明する。

第3図をC言語ソース・プログラム1としてマクロ処理部2に入力する。第3図にはマクロ処理される行がないため、マクロ処理部2の出力は入力と同じ第3図のC言語ソース・プログラムとなる。続いて第3図は、関数処理部4へ入力される。

関数処理部4は関数指定部3より第2図の関数指定テーブル8を受け取ると、関数指定テーブル8に書かれた“ラインナンバー7のsubr1”を入力された第3図へ捜しに行く。関数処理部4は第3図の7行目から“subr1”を見つけると、そこへ“subr1”の関数定義部分を挿入する。そして引数si2, si3は、それぞれ対応するi2, i3に変換し、“subr1”が返すリターン値はi1へ代入するように処理する。第3図には7行目以外

にも“subr1”が存在するが、第2図の関数指定テーブル8で指定されていないので処理を行わない。以上で指定された関数呼び出し部分についてインラインコード展開が行なわれたわけである。その結果、第4図がC言語中間プログラム6として作成される。

コンパイル処理部6は、第4図であるC言語中間プログラム6を入力し、オブジェクト・プログラム7を生成する。

以上のように本実施例によれば、関数指定部と関数処理部を設けることにより、例えば頻繁に呼ばれる関数のみインラインコード展開する関数として指定し、その他の関数はそのままにしておくと、生成されたオブジェクト・プログラムにおいて指定された関数は呼んだ側の関数の中で連続的に実行され、実行されるプログラムアドレスの不連続実行によるオーバーヘッドが発生せず、高速化が図れる。かつ、呼ばれる頻度の少ない関数はインラインコード展開しないため、記憶容量は少しの増加しか必要としない。

発明の効果

以上説明したように、本発明によれば、少しの記憶容量の増加で、関数の呼び出しによる不連続実行のオーバーヘッドを減少させ、プログラムの高速実行が図れ、その実用的効果は大きい。

4、図面の簡単な説明

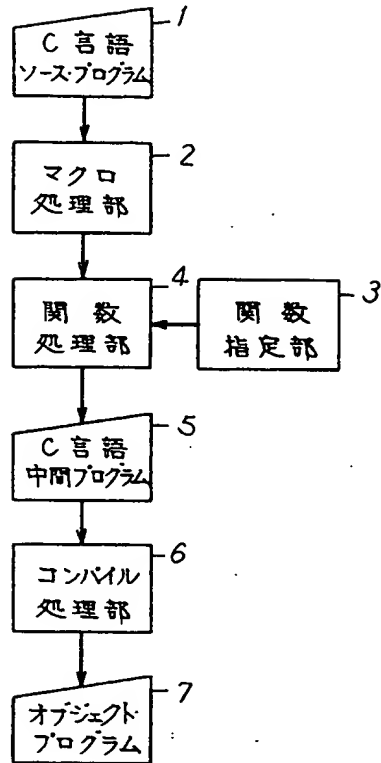
第1図は本発明における実施例のプログラム最適化処理方法の構成図、第2図は同実施例の関数指定テーブルの説明図、第3図は同実施例のC言語ソース・プログラムの説明図、第4図は同実施例のC言語中間プログラムの説明図、第5図は従来の実施例のプログラム処理方法の構成図、第6図は同実施例のC言語ソース・プログラムの説明図、第7図は同実施例のC言語中間プログラムの説明図、~~第7図は同実施例のC言語中間プログラムの説明図~~、第8図は従来の他の実施例のC言語ソース・プログラムの説明図である。

1, 9……C言語ソース・プログラム、2, 10……マクロ処理部、3……関数指定部、4……関数処理部、5, 11……C言語中間プログラム、

6, 12……コンパイル処理部、7, 13……オブジェクト・プログラム、8……関数指定テーブル。

代理人の氏名 弁理士 中 尾 敏 男 ほか1名

第 1 図



第 2 図

ラインNo	関 数 名
7	Subr1

第 3 図

```

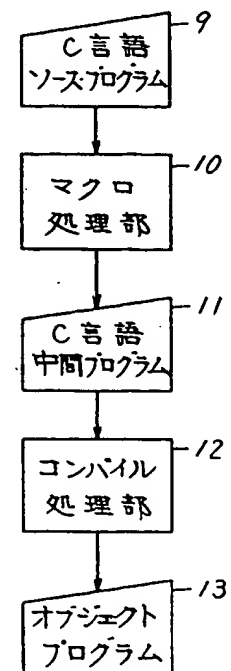
1  int    i1, i2, i3, flag;
2
3  main(){
4  int    c;
5
6      for( c=0; c<10; c++ )
7          i1 = subr1( i2, i3 );
8
9      if( i1 > i2 )
10         i1 = subr1( i2, i3 );
11     else
12         i1 = subr2( i2, i3 );
13 }
14
15 subr1( si2, si3 )  int    si2, si3;
16 {
17
18     return( flag & ( si2 << si3 ) );
19 }
20
21 subr2( si2, si3 )  int    si2, si3;
22 {
23
24     return( flag | ( si2 << si3 ) );
25 }
  
```

第 4 図

```

1  int    i1, i2, i3, flag;
2
3  main(){
4  int    c;
5
6      for( c=0; c<10; c++ )
7          i1 = flag & ( i2 << i3 );
8
9      if( i1 > i2 )
10         i1 = subr1( i2, i3 );
11     else
12         i1 = subr2( i2, i3 );
13 }
14
15 subr1( si2, si3 )  int    si2, si3;
16 {
17
18     return( flag & ( si2 << si3 ) );
19 }
20
21 subr2( si2, si3 )  int    si2, si3;
22 {
23
24     return( flag | ( si2 << si3 ) );
25 }
  
```

第 5 図



第 6 図

```
# define SUBR(x) (x & 3)

main(){
  int    x, y;
  y = SUBR(x);
}
```

第 7 図

```
main(){
  int    x, y;
  y = (x & 3);
}
```

第 8 図

```
main(){
  int    x, y;
  for( ; y != 0 ; )
    y = subr(x);
}

subr(sx)  int    sx;
{
  return(sx & 3);
}
```